SELES

An e-Voting System for Medium Scale Online Elections I

Garcia-Zamora, Claudia; Rodriguez-Henriquez, Francisco; Ortiz-Arroyo, Daniel

Published in:
Mexican International Conference on Computer Science

Publication date:
2005

Document Version
Early version, also known as pre-print

SELES: An e-Voting System for Medium Scale Online Elections

Claudia García-Zamora[1], Francisco Rodríguez-Henríquez[1], and Daniel Ortiz-Arroyo[2]
[1] Computer Science Section, Electrical Engineering Department
Centro de Investigación y de Estudios Avanzados del IPN
Av. Instituto Politécnico Nacional No. 2508, México D.F.
cgarcia@computacion.cs.cinvestav.mx, francisco@cs.cinvestav.mx
[2] Computer Science and Engineering Department
Aalborg University
Niels Bohrs Vej 8, 6700 Esbjerg Denmark
do@cs.aaue.dk

***Abstract*— Recent advances in communication networks and cryptographic techniques have made possible to consider on-line voting systems as a feasible alternative to conventional elections. Until today several protocols for electronic voting have been proposed, unfortunately only a few of them have been implemented in an end-to-end fully functional system. In this paper we present a secure electronic voting system for medium scale on-line elections (SELES). Our system efficiently implements a security communication protocol offering protection against double voting and others frauds while avoiding any private voting channel. SELES accomplishes all the standard properties of conventional voting systems, namely, accuracy, democracy, privacy, verifiability, simplicity, flexibility and double voting detection. Our system has been tested in a distributed and heterogeneous Internet network comprised by workstations, laptops and PDA nodes interacting through wired and wireless connections. Additionally, SELES has been designed to deal with communication failures, thus achieving a certain degree of robustness.**

Keywords: Electronic voting, Cryptography, Blind Signatures, DSA, RSA.

I. Introduction

In an electronic election system, privacy and security are mandatory features. However, it is not always obvious how to achieve these characteristics at a reasonable price, due to the fact that when an election process takes place, mechanisms that assure both, security and privacy may be too expensive for system administrators on one side, and inconvenient for users on the other.

Recent advances in communication networks and cryptographic techniques have made possible to consider on-line voting as a feasible alternative to conventional elections. Indeed, on-line electronic voting allows users to participate in an election no matter where they physically are at the moment of the voting process provided that they have a means of establishing a wired or wireless Internet connection to the system servers.

Additionally, an aggregated value of this kind of systems is its inherent privacy, since a voter can participate actively within an election process without being seen by anyone. Notice that it would be almost impossible to achieve this feature when using a traditional election system.

Creating an on-line voting system requires the use of robust security mechanisms that are relatively complex to design. Accordingly, the study of security mechanisms in electronic elections has received considerable attention in the last twenty years. As a result, a wide variety of e-voting cryptographic protocols have been proposed [1], [2], [4], [7], [10], [12], [15], [16]. Such protocols must satisfy a number of security requirements such as: vote accuracy, verifiability, voters' privacy, double voting detection, among others [12], [15].

Roughly speaking, those cryptographic protocols can be classified as the ones based on *Homomorphic functions*, and the ones based on *Blind signatures*.

The design of protocols based on *homomorphic functions*, requires rather complicated encryption schemes for hiding ballot's content in order to preserve voters' privacy [6], [16]. Those protocols include two phases: ciphering and voting. To implement these phases, several techniques such as shared secret keys and zero-knowledge proofs have been proposed. However, an important drawback of homomorphic protocols is that they tend to produce high communication overhead along with having high computational complexity in the vote counting phase.

Blind Signatures were proposed in 1983 by Chaum [1]. Protocols based on blind signatures hide voter's identity, but still make the actual content of a vote visible to the authority. Protocols based on blind signatures generally consist of a registration phase followed by a voting phase.

In this paper we present an RSA/DSA-based e-voting protocol for online elections which can be regarded as an improvement of the scheme proposed in [10]. As a means of testing the correctness of the said protocol, we implemented a fully-functional distributed e-voting system in Java. We give a detailed explanation of the e-voting system's architecture and its corresponding dataflow. Finally, we evaluate the performance achieved by our system and compare it with other

previously reported schemes.

The rest of this paper is organized as follows. In next Section, a summary of previous related work is outlined. In Section III, a description of the SELES protocol which is an improved version of Lin–Hwang–Chang's protocol is presented. Section IV discusses the rationale and implementation details behind the general architecture of the system developed. A performance and comparison evaluation of the system is presented in Section V. Finally, in Section VI some concluding remarks are drawn.

II. Related Work

Fujioka et al. [4], developed a practical voting scheme using blind signatures. In their proposal, each voter signs his/her vote with a secret key, and then sends it to the counting center through an anonymous channel. One disadvantage of this scheme is that the protocol is complex since the voting phase consists of two steps.

In 1997, L. Cranor and R. Cytron [2] proposed and implemented a protocol based on Fujioka's scheme called *Sensus*. The main difference between both schemes is that Sensus allowed users to vote in a single session, whereas Fujioka's proposal required two sessions. However, one disadvantage of these schemes is that the network traffic increases since the voter is required to send the same ciphered messages twice, making the protocol less efficient.

On the contrary, in Wen-Sheng et al. scheme [8], the network traffic is lower since every voter is allowed to send only a single anonymous message. Unfortunately, it has been shown that this scheme does not avoid vote duplication.

In 1998, Mu and Varadharajan [12] proposed two security schemes for electronic voting that addressed the issue of voter's privacy. The proposed schemes made use of RSA blind signatures along with ElGamal encryption scheme. These protocols were also able to detect vote duplicity.

In 1999, Karo and Wang [9] presented a security scheme for large scale electronic elections that did not used blind signatures. They suggested to utilize the HTTPS protocol to perform all transactions, instead of using an anonymous channel. However, the scheme was inefficient since six authorities were included in the model.

In 2001, Ray – Narasimhamurthi [13] designed a protocol that allowed anonymous voting through the internet. This protocol included three authorities and made use of digital certificates for voter authentication.

In 2003, Joaquim, Zúquete and Ferreira [7] presented a Java implementation of an electronic voting system called REVS. REVS implementation was based on the scheme proposed by DuRette [3], which was itself an improvement of the EVOX system described by Herschberg [5]. In REVS, every valid vote should contain t signatures from the n administrative entities, where: $t > \frac{n}{2}$.

In 2003, Chien et al. and Lin–Hwang–Chang [10] showed one weakness in the scheme proposed by Mu and Varadharajan: the possibility that a user could vote more than once without being detected. Lin–Hwang–Chang also presented in [10] an improvement to Mu and Varadharajan's protocol, adding a protection scheme against possible frauds based on the use of blind signatures. The proposed scheme did not require any special voting channel.

Finally in 2005, García-Zamora et al. [17] incorporated two amendments to the Lin–Hwang–Chang protocol in order to further improve its functionality. First, the usage of ElGamal encryption scheme in [10] was substituted by the Digital Signature Algorithm (DSA) scheme. By doing so, authors in [17] shown that independently of the random values that a voter and an authority server may choose, a vote will always be signed correctly before being sent to the voting server. Secondly, two extra encryption operations were added to the protocol dataflow in order to guarantee proper operation. A detailed description of this protocol is given in the next Section.

III. SELES voting protocol

SELES voting protocol consists of three phases: authentication, voting and counting. It considers the interaction of four entities, namely, voter (V), authentication server (AS), voting server (VS), and counting server (TCS). Cryptographic tools used by the protocol include digital certificates, time stamps, blind signatures and so on.

The notation that will be used to describe protocol's operation is as follows:

- V : voter
- AS: authentication server
- VS: voting server
- TCS: counting server
- t : time stamp
- q : DSA parameter, $2^{159} < q < 2^{160}$
- p : given l such that $0 \leq l \leq 8$, let p be a prime such that $2^{511+64l} < p < 2^{512+64l}$, with the property that q divides $(p-1)$, i.e, $q|(p-1)$.
- g : a generator for Z_p^*
- a : DSA private key $1 \leq a \leq q-1$
- α = $g^{(p-1)/q} \bmod p$
- y = $\alpha^a \bmod p$
- Cert: digital certificate issued by an authority
- $\{e_x, d_x\}, n_x$: a pair of RSA keys for user x, where $n_x = p_1 \times p_2$, p_1 and p_2 two large primes and $e_x \times d_x \bmod \phi(n_x) = 1$

In the rest of this Section, we summarize the main algorithm steps and dataflow performed during all three protocol phases.

A. Authentication

The authentication phase consists of three steps:

1) Voter chooses two blind factors b_1 and b_2, and two random numbers k_1 and a. Using these values together with the DSA parameters, the values y, z_1 and z_2 are generated in the following way:

$$\begin{aligned} y &= \alpha^a \bmod p, \\ z_1 &= [(\alpha^a \bmod p) \cdot (b_1^{e_{AS}})] \bmod n_{AS}, \\ z_2 &= [(\alpha^{k_1} \bmod p) \cdot (b_2^{e_{AS}})] \bmod n_{AS}. \end{aligned} \quad (1)$$

where p and α are public DSA domain parameters.

Then the voter sends,
$\{V, AS, Cert_V, t, z_1, z_2, [(z_1 \| z_2 \| t)^{d_V} \bmod n_V]\}$, to the AS.

2) AS validates V's identity by verifying the received signature $[(z_1 \| z_2 \| t)^{d_V} \bmod n_V]$ with the public key included in $Cert_V$. If the signature is valid, AS chooses a random number k_2 and stores it in the database as an identification of V. For this reason the value k_2 must be unique for each voter. Then AS generates z_3, z_4, z_5 and z_6 using the following procedure:

$$\begin{aligned} z_3 &= (k_2 \| t)^{e_V} \bmod n_V, \\ z_4 &= (z_1 \times AS)^{d_{AS}} \bmod n_{AS} \\ &= [(\alpha^a \bmod p) \times AS]^{d_{AS}} b_1 \bmod n_{AS} \\ z_5 &= (z_2 \times (\alpha^{k_2} \bmod p) \times AS)^{d_{AS}} \bmod n_{AS} \\ &= [(\alpha^{k_1} \bmod p) \times (\alpha^{k_2} \bmod p) \times AS]^{d_{AS}} b_2 \\ &\quad \bmod n_{AS} \qquad (2) \\ z_6 &= (z_2^2 \times (\alpha^{k_2} \bmod p) \times AS)^{d_{AS}} \bmod n_{AS} \\ &= [(\alpha^{2k_1} \bmod p) \times (\alpha^{k_2} \bmod p) \times AS]^{d_{AS}} b_2^2 \\ &\quad \bmod n_{AS} \end{aligned}$$

Finally, AS sends a reply message to V. Notice that in this message the values z_4, z_5 and z_6 are encrypted with V's public key, separately. Additionally, a timestamp t is added to the message. This is because it is not possible to encrypt all three values together, since the size will be 3072 bits and the size of n_V is only 1024 bits long. $\{AS, V, z_3, [(z_4 + t)^{e_V} \bmod n_V], [(z_5 + t)^{e_V} \bmod n_V], [(z_6 + t)^{e_V} \bmod n_V]\}$

3) The voter decrypts z_3 to get k_2. Additionally, he/she decrypts z_4, z_5 and z_6 using his/her private exponent d_V to decrypt the last three values in the reply message followed by the subtracting of t. Then, the blind factors are removed so that the signatures s_1, s_2 and s_3 can be generated as follows,

$$\begin{aligned} s_1 &= z_4 \times b_1^{-1} \\ &= [(\alpha^a \bmod p) \times AS]^{d_{AS}} \bmod n_{AS} \\ s_2 &= z_5 \times b_2^{-1} \\ &= [(\alpha^{k_1} \bmod p) \times (\alpha^{k_2} \bmod p) \times AS]^{d_{AS}} \\ &\quad \bmod n_{AS} \qquad (3) \\ s_3 &= z_6 \times b_2^{-2} \\ &= [(\alpha^{2k_1} \bmod p) \times (\alpha^{k_2} \bmod p) \times AS]^{d_{AS}} \\ &\quad \bmod n_{AS} \end{aligned}$$

B. Voting Phase

Voting phase dataflow is as described below.

1) Let us recall that VS needs to verify:
   - 2 voter's signatures of the vote m generated using DSA module q and;
   - 3 signatures module n_{AS}, which are the signatures that the AS previously provided to the voter.

Therefore, in the voting phase the voter proceeds to sign the ballot (m) using DSA and x_1 and x_2 as private keys. The voter is able to generate these values because he/she has already decrypted k_2. Notice that the two DSA signatures consists on the pairs (r_1, s_4) and (r_2, s_5). Hence, the first part of the signatures, namely r_1 and r_2, can be obtained as follows,

$$\begin{aligned} x_1 &= k_1 + k_2, \\ x_2 &= 2k_1 + k_2, \\ r_1 &= (\alpha^{x_1} \bmod p) \bmod q, \qquad (4) \\ r_2 &= (\alpha^{x_2} \bmod p) \bmod q. \end{aligned}$$

The second part of the signatures, namely s_4 and s_5, can be generated through the computation of the following equations:

$$\begin{aligned} s_4 &= x_1^{-1}(m + ar_1) \bmod q, \\ s_5 &= x_2^{-1}(m + ar_2) \bmod q. \qquad (5) \end{aligned}$$

Additionally the voter needs to compute the values l_1 and l_2 as,

$$\begin{aligned} l_1 &= [((\alpha^{k_1} \bmod p) \bmod n_{AS}) \times ((\alpha^{k_2} \bmod p) \\ &\quad \bmod n_{AS})] \bmod n_{AS}, \\ l_2 &= [((\alpha^{k_1} \bmod p)^2 \bmod n_{AS}) \times ((\alpha^{k_2} \bmod p) \\ &\quad \bmod n_{AS})] \bmod n_{AS}. \qquad (6) \end{aligned}$$

These last two values together with r_1 and r_2 are encapsulated taking advantage of the *Chinese Residue Theorem*. That is done with the goal of allowing VS to perform the corresponding verifications in the proper arithmetic (either modulus n_{AS} or modulus q).

$$\begin{aligned} pr_1 &= [(r_1 \times n_{AS}) + (l_1 \times q)] \bmod (n_{AS} \cdot q) \\ pr_2 &= [(r_2 \times n_{AS}) + (l_2 \times q)] \bmod (n_{AS} \cdot q) \end{aligned} \qquad (7)$$

Lastly, the voting ticket is generated in the following way:

$$Ticket = \{s_1, s_2, s_3, s_4, s_5, y, pr_1, pr_2, m\}$$

2) Voter V, sends his voting ticket to VS. VS performs a total of 5 signature verifications needed for ticket validation. The first 3 verification equations are as follows

$$\begin{aligned} (AS \times y) \bmod n_{AS} &\stackrel{?}{=} s_1^{e_{AS}} \bmod n_{AS} \\ (AS \times pr_1 \cdot q^{-1}) \bmod n_{AS} &\stackrel{?}{=} s_2^{e_{AS}} \bmod n_{AS} \\ (AS \times pr_2 \cdot q^{-1}) \bmod n_{AS} &\stackrel{?}{=} s_3^{e_{AS}} \bmod n_{AS} \end{aligned}$$

Notice that in virtue of the *Chinese Residue Theorem* we have that $pr_1 \cdot q^{-1} = l_1$ and $pr_2 \cdot q^{-1} = l_2$, where q^{-1} is defined as the multiplicative inverse of q modulus n_{AS}. Similarly r_1 and r_2 can be recovered by computing

$pr_1 \cdot n_{AS}^{-1} = r_1$ and $pr_2 \cdot n_{AS}^{-1} = r_2$, where n_{AS}^{-1} is defined as the multiplicative inverse of n_{AS} modulus q.
In order to verify the DSA signatures we use the following procedure,

$DSAverifier(r, s)$ {
$w = s^{-1} \bmod q$
$u_1 = w \cdot m \bmod q$
$u_2 = r \cdot w \bmod q$
$v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$
return v
}

Then the last two signatures s_4 and s_5 can be verified by proving that the following equations hold:

$$r_1 \stackrel{?}{=} DSAverifier(r_1, s_4)$$
$$r_2 \stackrel{?}{=} DSAverifier(r_2, s_5)$$

3) If all five signatures are correctly verified, VS will accept and store the ticket sent by the voter as a valid one. Once that the voting election process has been completed VS sends all valid votes that were received to TCS over the communication network.

C. Counting phase

TCS must receive all valid tickets from the voting servers. Additionally, TCS must identify all tickets that are identical and count them only once. These actions will guarantee a final tally equal to the total number of the valid votes received during the elections.

In this phase it is possible to detect malicious voters that may have sent two or more tickets with different votes. In order to perform the so-called *double voting detection*, we consider the fact that a given voter uses the same key to sign different votes. Therefore, TCS will receive at least two tickets with the following form:

$$B_1 = \{s_1, s_2, s_3, s_4, s_5, y, pr_1, pr_2, m\},$$
$$B_2 = \{s_1, s_2, s_3, s_4', s_5', y, pr_1, pr_2, m'\}.$$

With the information contained in these two tickets, TCS is capable of identifying the voter who sent these ballots, by computing the following equations:

$$\begin{aligned} x_1 &= \frac{m' - m}{s_4' - s_4} \bmod q, \\ x_2 &= \frac{m' - m}{s_5' - s_5} \bmod q, \\ k_1 &= x_2 - x_1, \\ k_2 &= x_1 - k_1. \end{aligned} \tag{8}$$

As it was mentioned previously, all k_2 values assigned to each voter are stored in the database of *AS*. In this way *TCS* can request to *AS* the name of the voter which is associated to the computed value k_2, thus identifying the identity of the malicious voter.

IV. Implementation

SELES has been designed and implemented for medium scale electronic elections, i.e., limited to several tenths of thousands voters. The limit on the number of voters is mainly imposed by the WEB server utilized in the implementation. SELES functionality is based on the *Client–Server* paradigm. At the same time, SELES makes use of several cryptographic tools and techniques which provide a correct and secure performance.

Voter's functionality is provided by an application that we have called *Voter Application* which was specifically designed for this purpose. This application can run in a variety of platforms such as Personal Computer, Laptops, or even a Personal Digital Assistant (PDA).

Due to the fact that votes are emitted online, eligible users of a given election can participate via either a wired or a wireless Internet connection. The system's general architecture envisioned for SELES is shown in Fig.1.

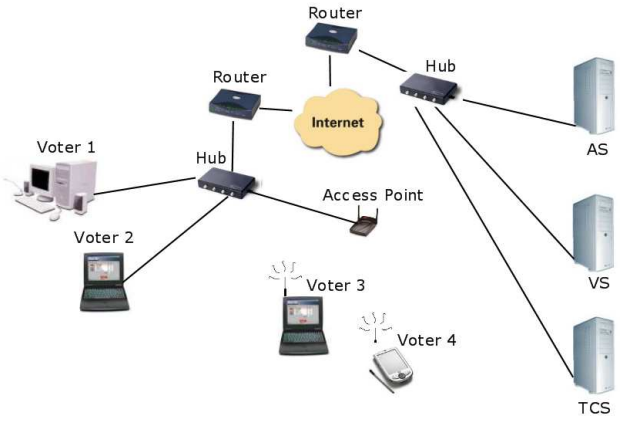

Fig. 1: System's General Architecture

Those participants working on high performance computers (such as workstations or laptops) may cast their votes without using a local application. Instead, all voter's actions will be accomplished by using *signed applets* previously downloaded by the voter during his/her registration in the voting process.

Voters who wish to cast their ballots from a PDA device, must download and install a *Voter Application*. We were forced to develop that application because today PDA's Internet browsers support standard *applets* only and almost none of them can deal with *signed applets*.

Fig. 2 shows a top-down layer model used for implementing each one of the entities considered by our system. As it was mentioned, SELES protocol defines three authorities, namely, AS, VS y TCS. Additionally, a Certificate Authority (CA) is also required. This authority needs to participate in a preliminary phase, generating the RSA public/private key pair for each participant. In this work, we considered that all participants (voters and authorities) have a 1024-bit RSA type of public/private key pair. Furthermore, the CA is also

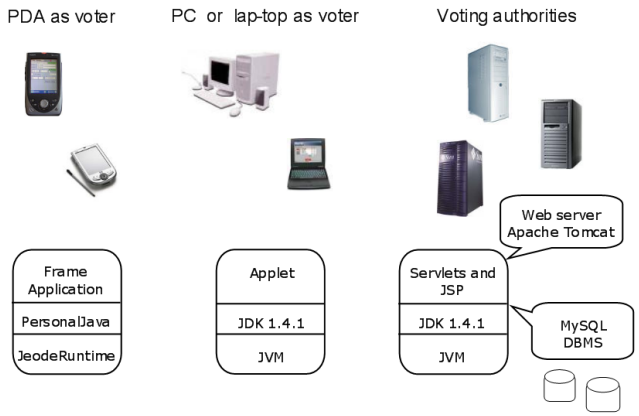

Fig. 2: SELES Top-Down Layer Model

responsible of issuing the digital certificates corresponding to each public key.

Once again and due to PDAs computing power limitations, we faced some compatibility problems when trying to read/access the private keys generated by our own CA application *ACERPAM*. The reason for this problem is that unfortunately crucial Java libraries are not included yet in the *Personal Java* package, which is the typical Java virtual machine available for PDA devices. Due to that, those users wishing to generate and transmit a vote from a PDA device must utilize public/private keys and certificates generated by a cryptographic tool called *keytool* which is a utility included in JDK 1.2.

A. Authentication Server

The Authentication Server (AS) is responsible for authenticating all registered users wishing to vote. SELES considers that a legitimate user is the one that has been correctly preregistered in the AS election database. Fig. 3 shows the dataflow between a voter and the AS during the *authentication phase*.

The authentication server is the entity in charge of receiving users' digital certificates along with their personal data. Afterwards, AS verifies the public identifier and the signature it just received. Next step is to check whether the prospective voter has been previously authenticated or not, i.e., if a blind signature was previously granted to him/her or not. If the answer is yes, then AS recovers the user's unique identifier value k_2 from its database. On the other hand, if this is the first time that the voter is establishing contact with the AS, a unique value k_2 is assigned to him and stored in the database so that future identification of that user is possible. Finally, AS blindly signs the corresponding values according to SELES security protocol described in last Section. AS encrypts the blind signatures along with the value k_2 just generated and the message time stamp. This information is encapsulated in a message that will be sent to the voter later.

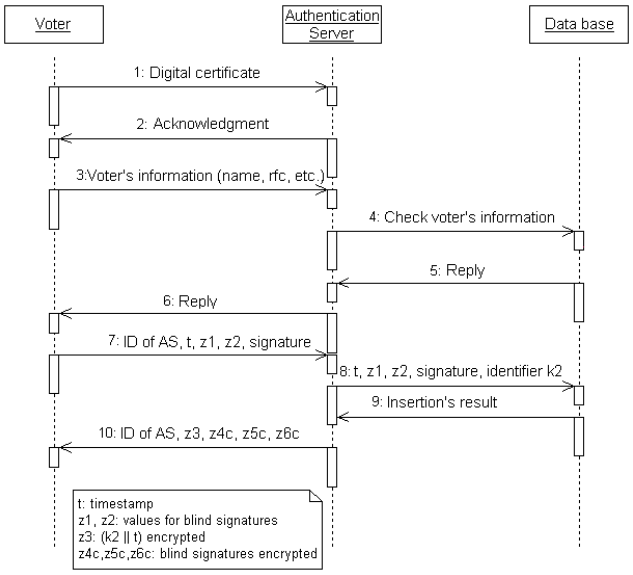

Fig. 3: Voter-AS UML Sequence Diagram

B. Voting Server

The Voting Server (VS) has the responsibility of accepting participant's ballots for a given election process. Received ballots must contain:

1) *vote selection* The candidate selection chosen by the participant,
2) *Public Parameters* Which are needed to verify two digital signatures included in the vote and,
3) *Five signatures* According to SELES Security protocol, three of those verified signatures are RSA signatures while the other two are DSA signatures.

Fig. 4 shows the dataflow between a voter and the VS during the *voting phase*. After the VS receives the ballot, it gets AS public key which is required for verifying the first three RSA signatures. If all the three RSA signatures get validated, then the remaining two DSA signatures are verified. If all five signatures are valid, then an automatic response message is generated where the VS let the voter know that his/her vote was correctly generated. However, if any one of the five signatures fails, the response message informs that the ballot has not been accepted.

C. Ticket Counting Server

The Ticket Counting Server (TCS) receives from VS all registered tickets whose five signatures have passed the verification step. Fig. 5 shows the corresponding dataflow between VS and TCS. Obviously, TCS' main responsibility is of counting all tickets in a fair and exact manner.

To do that, TCS first stores all received ballots in its database. Then, it proceeds to identify all identical ballots so that they will be considered only once during the final counting.

The same procedure is instrumented for detecting duplicated tickets that may have being sent by malicious voters. In

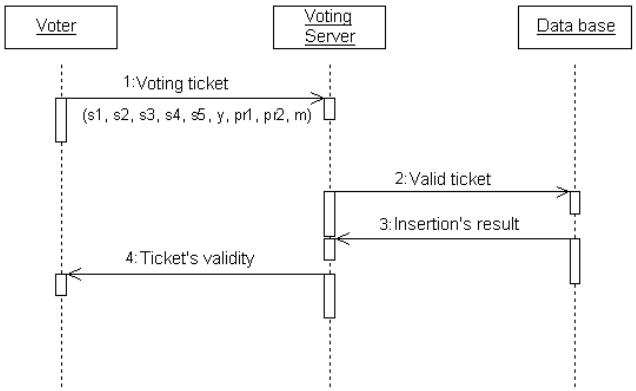

Fig. 4: Voter-VS UML Sequence Diagram

the case that some suspicious tickets are found, TCS must obtain the corresponding identifiers and report them back to AS. Afterwards, AS will extract the cheaters' identity corresponding to those reported identifiers. Fig. 6 shows the dataflow between TCS and AS.

Only after having analyzed all received tickets, TCS proceeds to perform the final counting followed by the publication of the official results. By performing these actions, SELES system guarantees a final tally equal to the total number of the valid votes received during the election process. Finally and in an effort to bring transparency to the system, TCS publishes a detailed list of all valid and invalid tickets that were received during the process.

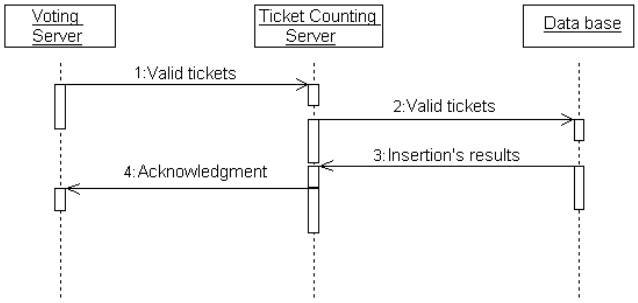

Fig. 5: VS-TCS UML Sequence Diagram

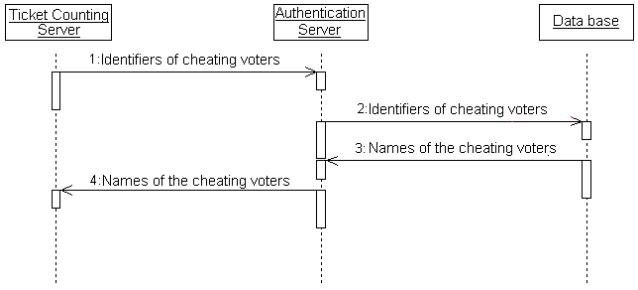

Fig. 6: TCS-AS UML Sequence Diagram

D. Voter

In order to be able to cast a vote, a participant must complete two phases;

1) *Authentication Phase*: During this phase the voter sends his/her digital certificate and personal data to AS. To do so, the voter first obtains AS public key and prepares and signs the message that AS must blindly sign. When the voter receives the response message from AS it needs to parse it so that the value k_2 generated by AS, (let us recall that this value is the voter's unique identifier) can be obtained along with the three blind signatures. Thereafter, the voter can remove the blind factor and store the parameters contained in the response message.
2) *Voting Phase*: At the beginning of this phase, the participant fills out a ticket with his/her own electoral candidate selections. Afterwards the participant must digitally sign the ticket using the DSA cryptosystem and send it to the VS. Then, VS will proceed to check whether that ticket is valid or not. In case of a positive result, VS sends an OK message back to the voter, otherwise the voter receives an error message. If everything goes well the voter hashes the ticket and stores it together with ticket's relevant parameters.

It is worth to mention that when election's final tally is published, any legal voter can make sure that his/her vote was actually counted. This check can be done by any legal voter in a simple manner: he/she just need to check that the hash of his/her ticket is included among all tickets listed by TCS.

In order to prevent typical package losses over Internet, SELES makes use of time stamps and safe storage of ticket' relevant parameters. In this way SELES achieves some degree of robustness.

E. Implementation Details

All three authorities were implemented using Servlets and JSP's. The Voter application was coded in Java using J2SDK version 1.4.1_02.

The PDA version of the *Voter application* was written using the execution environment Insignia JeodeRuntime, which is fully compatible with PDA Personal Java. The PDAs used in this work were: HP iPAQ Pocket PC model h5500, and a SHARP model Zaurus SL-5500, with operating system Windows and Linux, respectively. Let us remark that exactly the same *Voter application* code runs in both PDA devices in spite of the platform differences.

The WEB server utilized was **Apache Tomcat** version 5.0.2. Finally, the database management was coded using MySQL version 1.4 and **MySQL Control Center** 0.8.9-beta as graphic user interface for MySQL.

V. Evaluation

A. Functionality

The scheme proposed in this paper has the following properties:

TABLE I: Desired Properties

SCHEME	ACCURACY	DEMOCR.	PRIVACY	VERIFIC.	CONVEN.	FLEXIB.	DV-DET.
Schoenmakers [14]	√	√	√	√	√	×	×
Fujioka et al. [4]	√	√	√	√	×	√	×
Sensus [2]	√	√	√	√	√	√	×
Karro et al. [9]	√	√	√	√	√	√	×
REVS [7]	√	√	√	√	√	√	×
SELES	√	√	√	√	√	√	√

1) *Accuracy*: SELES security protocol guarantees that only valid tickets will be counted in the final results, since VS will not accept any ticket that does not contain the 3 signatures issued by the AS. Moreover, a procedure has been included in the protocol to detect whenever two or more valid tickets are issued by the same voter.
2) *Democracy*: This property is achieved because AS will verify the identity of a voter, checking that he/she meets all requirements to participate in an election.
3) *Privacy*: With the use of blind signatures it is guaranteed that given a vote, it is not possible to determine whose person voted.
4) *Verification*: Since every ticket issued contains 5 digital signatures (3 realized with the keys chosen by a voter and 2 user keys generated between the voter and the AS), and the unencrypted vote (m), voters may verify that their vote was correctly counted whenever the results of an election are published. This is possible because TCS not only publishes final results, but also all the valid tickets that were received.
5) *Convenience*: Voters are capable of finishing the voting process in short time, during a single session and with minimum equipment.
6) *Flexibility*: The system is flexible because it is based on blind signatures, which allows the use of several formats for the voting ticket.
7) *Detection of 2 or more votes issued by a single voter*: SELES protocol is able to detect whether a voter has issued two or more votes, additionally of being capable of knowing the identity of a malicious voter.

B. Comparison

Table I shows a comparison among some of the most well known e-voting schemes. Properties considered include: accuracy, democracy, privacy, verification, convenience, flexibility and double voting detection, respectively.

Table II shows a comparison between some of the protocols listed in Table I, in terms of the total number of keys pairs, passwords and authorities required by those protocols. Additionally, we compare the number of times that a vote is sent through the network in the voting phase. In Table II, N is the number of votes sent in a given election. In the case of the REVS protocol, t corresponds to a value greater than $\frac{i}{2}$, where i is the number of *Administrative* entities that are participating in an election. Finally in the case of Karro's protocol, k is the value of key pairs that the *Counting* authority decides to issue. The authority must issue a number of encrypted tickets greater than the number of registered voters, using different keys.

TABLE II: Comparative Table

SCHEME	KEYS	PASSWORD	AUTH.	TRANSM.
Sensus [2]	$1 + N$	1	3	$3N$
Karro et al [10]	$1 + kN$	0	6	$3N$
REVS [7]	$2 + iN$	t	$3 + i$	$2N$
This Protocol	$1 + 2N$	1	3	N

The number and size of exchanged messages transferred during both, authentication and voting phase are summarized in Table III

TABLE III: Aproximate size of messages

PHASE	MESSAGE 1	MESSAGE 2
Authentication	19.5Kb	4.5Kb
Voting	06.7Kb	—

A summary of the cryptographic operations performed all along the election process is shown in Table IV

TABLE IV: Cryptographic Operations

PHASE	V	AS	VS
Authentication	1 RSA sign. 2 RSA encryp.	1 RSA verif. 4 RSA encryp. 3 blind sign.	
Voting	4 RSA decryp. 2 DSA sign.		3 RSA verif. 2 DSA verif.
TOTAL	9 operations	8 operations	5 operations

Finally in Fig 7 we show SELES execution time needed for election's tally computation.

VI. Conclusions

In this contribution we presented the design and implementation of SELES, an end-to-end electronic election system solution. SELES security and privacy features are provided by a secure communication protocol which is an improved version of the Lin-Hwang-Chang's scheme [10]. SELES' performance was tested and measured using a client-server model implemented over a heterogeneous distributed system comprised of workstations, laptops and PDA devices interacting through wired and wireless Internet connections. A comparative analysis among reported e-voting systems shows that SELES is a competitive option able to fulfill all desirable properties of a secure election system, such as accuracy, democracy, verification, convenience, flexibility and detection of double voting. Furthermore, our experiments also show that SELES is able to obtain a final exact tally of up to five thousand votes in less than 140 Seconds.

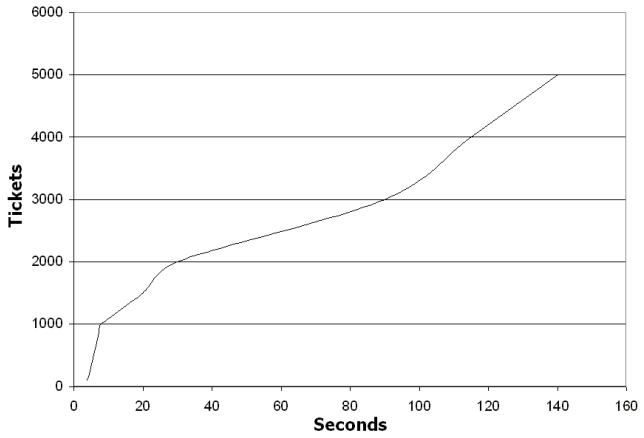

Fig. 7: Time required for Tally Computation during the Voting Phase

REFERENCES

[1] D. Chaum, "Blind signatures for untraceable payments," *Advances in Cryptology, CRYPTO'82,* (1982): 199–203.

[2] L. F. Cranor y R. K. Cytron, "Sensus: A security-conscious electronic polling system for the Internet," *Proceedings of the Hawaii International Conference on System Sciences,* **vol.**(3), (Wailea, Hawaii, 1997): 561–570.

[3] B. DuRette, "Multiple Administrators for Electronic Voting," Tesis de Licenciatura, *Massachussetts Institute of Technology,* (1999).

[4] A. Fujioka, T. Okamoto, y K. Ohta, "A practical secret voting scheme for large scale elections," *Advances in Cryptology – AUSCRYPT '92,* **vol.**(718), (Lecture Notes in Computer Science. Springer – Verlag, Berlin, 1993): 244–251.

[5] M. Herschberg, "Secure electronic voting over the word wide web," Tesis de Maestría, *Massachussetts Institute of Technology,* (1997).

[6] Kenneth R. Iversen, "A cryptographic Scheme for Computerized General Elections," *Advances in Cryptology, CRYPTO'91,* **vol.**(576), (Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1992): 405–419.

[7] Rui Joaquim, André Zúquete y Paulo Ferreira, "REVS - A Robust Electronic Voting System," *Proceedings of IADIS International Conference e-Society 2003,* (Lisbon, Portugal, 2003): 95–103.

[8] Wen-Shenq Juang y Chin-Laung Lei, "A secure and practical electronic voting scheme for real enviroments," *IEICE Transactions on Fundamentals,* **vol.**(E80–A, no. 1), (1997): 64–71.

[9] Jared Karro y Jie Wang, "Towards a Practical, Secure, and Very Large Scale Online Election," *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99),* (Scottsdale, AZ. Diciembre, 1999): 161–169.

[10] Iuon-Chung Lin, Min-Shiang Hwang y Chin-Chen Chang, "Security enhancement for anonymous secure e-voting over a network," *Computer Standards & Interfaces,* **vol.**(25, no. 2), (2003): 131–139.

[11] Shu-Chen Lin, "A Study on Proxy E-Voting Schemes," Tesis de Maestría, *Chaoyang University of Technology department of information management,* (Mayo, 2004).

[12] Yi Mu y Vijay Varadharajan, "Anonymous secure e-voting over a network," *Proceedings of the 14th Annual Computer Security Applications Conference,* (IEEE Computer Society, 1998): 293–299.

[13] Indrajit Ray, Indrakshi Ray y Natarajan Narasimhamurthi, "An anonymous electronic voting protocol for voting over the internet," *Proceedings of the Third International Workshop on Advanced Issues of E–Commerce and Web–based Information Systems,* (San Juan, CA, 2001): 188–190.

[14] Berry Schoenmakers, "A simple publicly verifiable secret sharing scheme and its Application to the electronic voting," *Advances in Cryptology, CRYPTO'99,* **vol.**(1666), (Lecture Notes in Computer Science, 1999): 148–164.

[15] Indrajit Ray, Indrakshi Ray and Natarajan Narasimhamurthi, "An anonymous electronic voting protocol for voting over the internet," *Proceedings of the Third International Workshop on Advanced Issues of E–Commerce and Web–based Information Systems,* (San Juan, CA, 2001): 188–190.

[16] Berry Schoenmakers, "A simple publicly verifiable secret sharing scheme and its Application to the electronic voting," *Advances in Cryptology, CRYPTO'99,* **vol.**(1666), (Lecture Notes in Computer Science, 1999): 148–164.

[17] Claudia Patricia García-Zamora, Francisco Rodríguez-Henríquez and Daniel Ortiz-Arroyo,"An Efficient and effective e-voting protocol for online elections" Technical Report CINVESTAV-IPN, 17 pages, 2005. available at: http://delta.cs.cinvestav.mx/∼francisco/Repository/techreport2005.pdf